ペトリネットシミュレータを用いたロボットの制御

安田元一*・野崎宏之**

Control of Industrial Robot Systems using a Petri Net Simulator

YASUDA Gen'ichi and NOZAKI Hiroyuki

This paper deals with the development of a simulator and a controller for discrete event system control of robotic manufacturing systems using Petri nets. The simulator executes the Petri net model representing the given robotic task on the color display for visual examination of the system's behavior. The simulator is applied to the analysis and synthesis of discrete event system control and the simulation results can be used for the control algorithm of the developed controller. The controller is programmable by Petri nets directly, and its processing speed is higher than that of usual programmable logic controllers programmed by relay diagrams or logic diagrams. Consequently, the consistent process of system synthesis, analysis and control by Petri net technique can be realized by combining the simulator and the controller.

1．緒　言

自動車の生産ラインを始めとするさまざまな産業分野にロボットや知的作業機械が導入され，それらの制御にコンピュータを用いる例も増加し，また生産システムの知能化に対する要求も高まってきている。さらに，多様な製品を効率的に生産することができるFMS（フレキシブル生産システム）やセル生産システム，および，製品の企画から流通，サービスまでの情報と活動をネットワークで統合するCIM（コンピュータ統合生産）ではソフトウェアの開発や，保守，変更が単純かつ迅速に行えることが望まれている。このような課題を解決するためには，生産システムそのもののあり方を考え直して，ロボットなどの構成要素を自律的な知的エージェントとし，要素が知的に判断することで通信し協調しながら作業を進めていくという考え方が，中央コンピュータのソフトウェアの負担が軽くなるという観点からも有効と考えられる。知能化された要素システムにより，ローカルな情報はローカルに処理し，抽象化，本質化された情報だけを上位の意思決定機構に伝達すれば中央と各要素システムとの間の通信の負担も軽減される。

大規模かつ複雑な生産システムの構造は，あるシステムはいくつかのラインによって，あるラインはいくつかのステーションによって，あるステーションはいくつかのマシンによって構成されているというような階層構造になっている。この

* 情報学部知能情報学科教授
**不動技研工業㈱
2008年9月30日受付

ような生産システムの制御仕様の記述において非同期・並列システムの表現方法であるペトリネットの概念に基づく手法が有効である[1),2)]。その際，システムの構造に対応して階層的にモデル化していくと仕様設計が容易で，かつ見通しがよくなる。また，ペトリネットに基づいて表現されたモデルに従って実システムを動作させることにより，システムの制御や管理にも応用が可能である。

現在，産業用だけでなく，さまざまな方面でロボットが使用されている。しかし，ロボットの作業（行動）を的確に表現しプログラムする方法が確立していない。ロボットの作業も一つの離散事象システムであるから，ペトリネットを適用できるはずである。この際にロボットの作業をどのようにペトリネットに対応づけるか，また，ロボットの作業には多くの制御情報をともなうため，コントローラではどう処理するかが問題となる。さらに，ロボットのプログラムは一般に複雑な副作用をもち，実行時間も長くなるため，オンライン修正と故障時の初期化が速やかに行えるプログラミングシステムが望まれる。

本研究は，ロボット，コンベアなどの知能化機器から構成される生産システムを対象として，ペトリネットに基づく離散事象システムとしてのモデル化とシミュレータの開発を行い，シミュレータからの機器の状態監視とコマンド実行の機能を付加することによりコンピュータを用いた統合制御システムを構築し，実験によりシステムの動作確認を行ったものである。

2．ペトリネットによる離散事象システムの表現

生産システムにおいては，素材や部品等が工程の各段階を順に進みながら各段階固有の操作を受ける。その際必ず，各工程の切れ目を通過し次の工程に移る，という不連続性が本質的に存在する。このようなシステムの中では，加工や組立てのある工程で「処理中」という状況は，「処理終了」という事象の発生で途切れ，そして次の状況に移る。状況は事象が起きるための条件と見ることもできる。このように状況と事象の交互の接続で活動しているシステムを離散事象システムと呼んでいる。このような離散事象システムは非同期性，順序性，並列同時進行性，競合などの性質をもち，生産システムのほか，コンピュータ内部のOSの情報処理や通信プロトコルなど，大規模・複雑化した多くのシステムにその例がみられる。離散事象システムを設計し合目的的に制御するには，概念レベルから詳細レベルまでシステムの性質を正確に表現でき，関係者間の情報交換やシステムの解析・設計に有効な表現方法が必要である。

ペトリネットはプレース，トランジション，アーク，トークンという要素によって構成され，事象をトランジションで，状況をプレースで表す。アークは状況と事象の接続関係を示す。トークンは状況が保持されていることを示すものでプレース内におかれる。

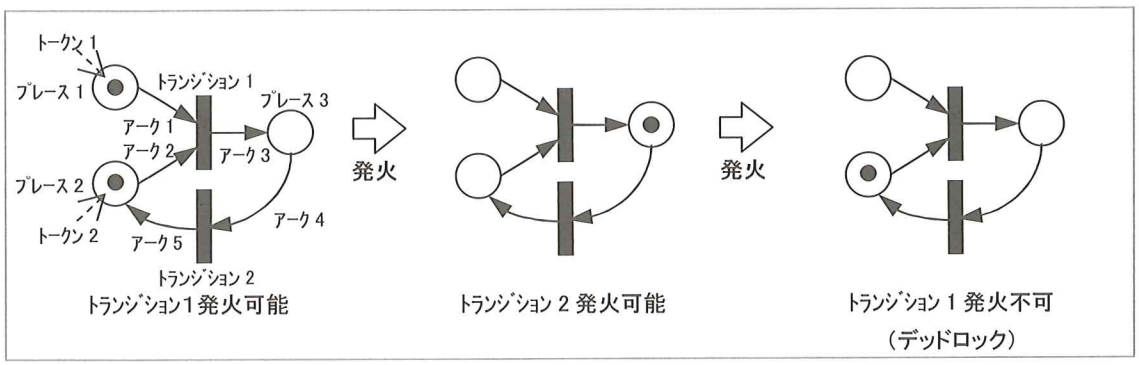


図1　基本的な発火則

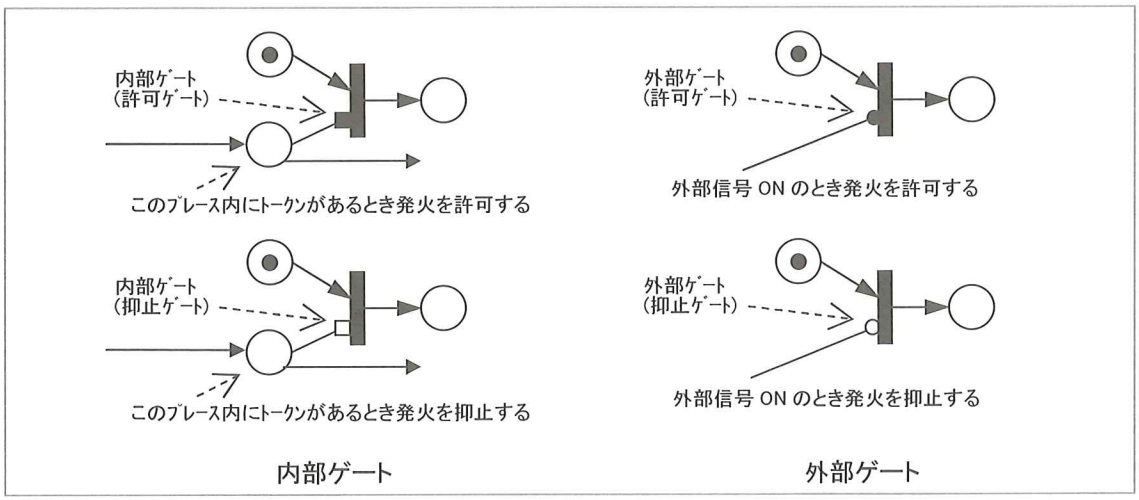


図２　ゲート枝の発火則

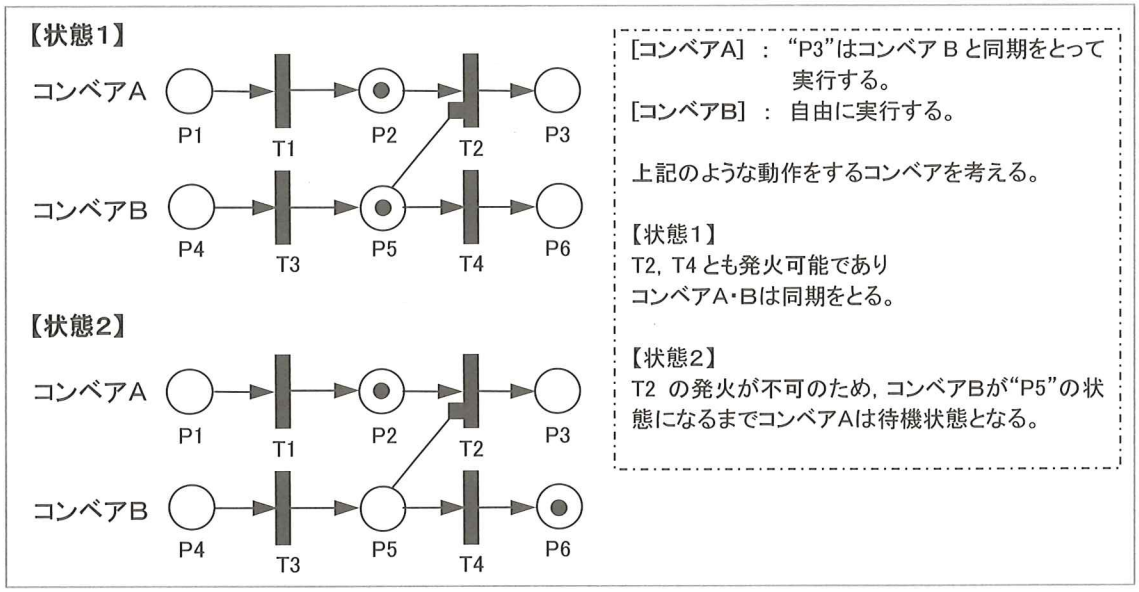


図３　内部ゲート使用例

ペトリネットにおける事象の発生と状況の移りかわりはトランジションの発火規則で示される。あるトランジションの入力側の全てのプレースがトークンを持ち，かつ出力側の全てのプレースにトークンがない時，そのトランジションは発火可能である。発火可能トランジションが発火すると，入力トークンは消滅しその出力プレースにトークンが発生する。ペトリネットの基本的な発火の例を図１に示す。

本研究では実際の機器の制御を考慮し，ペトリネットの内部または外部からトランジションの発火を制御するためのゲート枝をペトリネットの要素として用いた[3]。ゲート枝には内部ゲートと外部ゲートがあり，内部ゲートは，あるプレース内のトークンの有無を条件とする。トランジションの入力側に接続されるプレースとそのトランジ

ションの内部ゲートとして設定されるプレースとの違いは，トランジションの発火によりプレース内のトークンが移動するか否かである。すなわち内部ゲートとして設定されるプレース内のトークンの有無はそれを条件とするトランジションの発火に左右されないため，並行して進行する別システムの“活動”状態を判定し発火の条件とするような使い方が考えられる。外部ゲートは外部信号のON/OFFを条件とする。システム外部からの強制的なシステム停止・待機などの使い方が考えられる。さらにそれぞれのゲート枝は許可ゲートと抑止ゲートに分類され，許可ゲートは，内部ゲートにおいてはプレース内にトークンがある場合に条件成立，外部ゲートにおいては外部信号ONの場合に条件成立となる。抑止ゲートは，内部ゲートにおいてはプレース内にトークンがない場合に条件成立，外部ゲートにおいては外部信号OFFの場合に条件成立となる。ゲート枝の発火例を図2に示す。また，内部ゲートの具体的な使用例を図3に示す。

3．ペトリネットの内部表現

ペトリネットは図的表現であり，これを計算機で取り扱う際には情報化された内部表現が必要となる。ここで図4に示す接続行列を考える。これはトランジションとプレースの接続関係を示すものであり，要素が“1”であればトランジションの出力側，“－1”であればトランジションの入力側にプレースが接続され，“0”のときはトランジションとプレースは未接続であることを示す。ところがペトリネットは順序性（前後関係），並列性より接続関係をもつトランジションとプレースは限られている。つまり接続行列において要素が“0”であるものは多い。これでは無駄が多く処理に要する時間も大きくなる。このため接続行列で表したものを新たに接続関係表としてまとめる。これはトランジションを中心に考え，そのトランジションに接続されているプレース番号に入力側，出力側の区別をつけて表したものである。接続関係表とその他の情報化した内部表現であるゲート条件表，アービタ表，マーキング表の具体例を図5に示す。ゲート条件表はトランジションを中心に考え，そのトランジションのゲート枝の番号を内部ゲート，外部ゲート，許可ゲート，抑止ゲートの区別をつけて表す。アービタ表は複数のトランジションが接続されているプレース，つまり競合プレースに対して接続されているトランジション番号を表す。アービタとは競合プレースに接続された複数個のトランジションが発火可能なとき，一定の順序と割合で発火の権利を割り当てて調停するものである。マーキング表は各プレースにトークンが存在するか否かを表す。

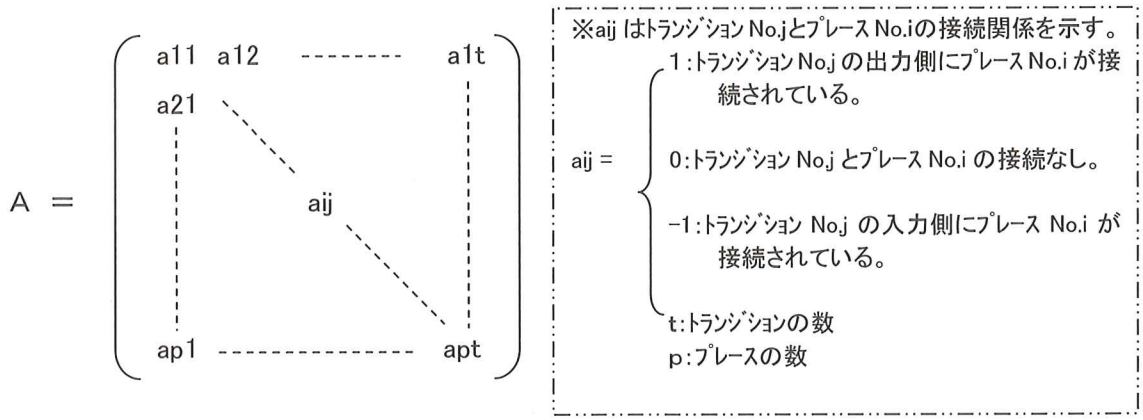


図4　接続行列

【接続関係表】

トランジション No.	プレース No.
1	−1 , 3
2	−5 , 4
⋮	⋮
⋮	⋮
t	−(p − 1) , p

例）
トランジション No.1 に関してプレース No.1 が，出力側にプレース No.3 が接続されている。

{ プレース No. が正：トランジションの出力側接続
{ プレース No. が負：トランジションの入力側接続

【ゲート条件表】

トランジション No.	ゲート No.
1	−E1
2	I5
3	E2
⋮	⋮
t	

例）
トランジション No.1 に外部ゲート(抑止ゲート)の条件あり。
トランジション No.2 に内部ゲート(許可ゲート)の条件あり。
トランジション No.t にゲート条件なし。

{ E：外部ゲート　　{ 正：許可ゲート
{ I：内部ゲート　　{ 負：抑止ゲート

なお，外部ゲートの No.は通し番号で，内部ゲートの No. は関連するプレース No.と合わせる。

【アービタ表】

通し番号	トランジション No.
1	2 , 3 , 4
2	1 , 5
⋮	⋮
⋮	⋮
N	(t − 2) , t

例）
トランジション No.2, 3, 4 が接続されている競合プレースが存在する。

【マーキング表】

プレース No.	トークン(1 or 0)
1	1
2	0
3	0
⋮	⋮
p	1

例）
プレース No.1 にはトークンが存在する。
プレース No.2 にはトークンが存在しない。

{ 1：トークンあり
{ 0：トークンなし

図５　情報化した内部表現の具体例

4．ペトリネットシミュレーションの方法

システムのシミュレーションは，ペトリネットの発火規則に基づいてトークンを移動することにより行う[4]。ペトリネットモデルを描画することから始めるため，トランジション，プレース，アーク，ゲート枝，トークンを個別に描画できるようにする。描画完了後に情報化した内部データに変換し，それを利用して各トランジションが発火可能か否かを記憶するトランジション情報表を作成する。これはトランジションの番号ごとに“１”

（発火可能），“0”（発火不可）で表したもので，シミュレーションの進行とともに変化していく。このトランジション情報表を用いることでシステムの停止状態が外部ゲートの成立を待っているためなのか，それともデッドロックなのかを判定することができる。また，グラフィック表示においてその挙動が分かりやすいことは重要であるので，単にトークンの移動をするだけでなく発火可能なトランジションの色替えをして明示するようにした。シミュレーションのフローチャートを図6に示し，以下に各処理について説明する。

(1)　モデル描画

シミュレータの機能としては欠かせないペトリネットモデルをグラフィック作成・表示する。ペトリネットの要素であるトランジション，プレース，アーク，ゲート枝，トークンを個別に描画し，アークによるトランジションとプレースの接続，ゲート枝のトランジションへの配置，トークンのプレースへの配置等，後の内部データに関わる重要な事項に関して不具合があるときはエラーメッセージを出すようにする。また，描画時にそれぞれのグラフィック内座標データを記憶させることで，グラフィックデータの色替え，修正，保存，展開が可能である。

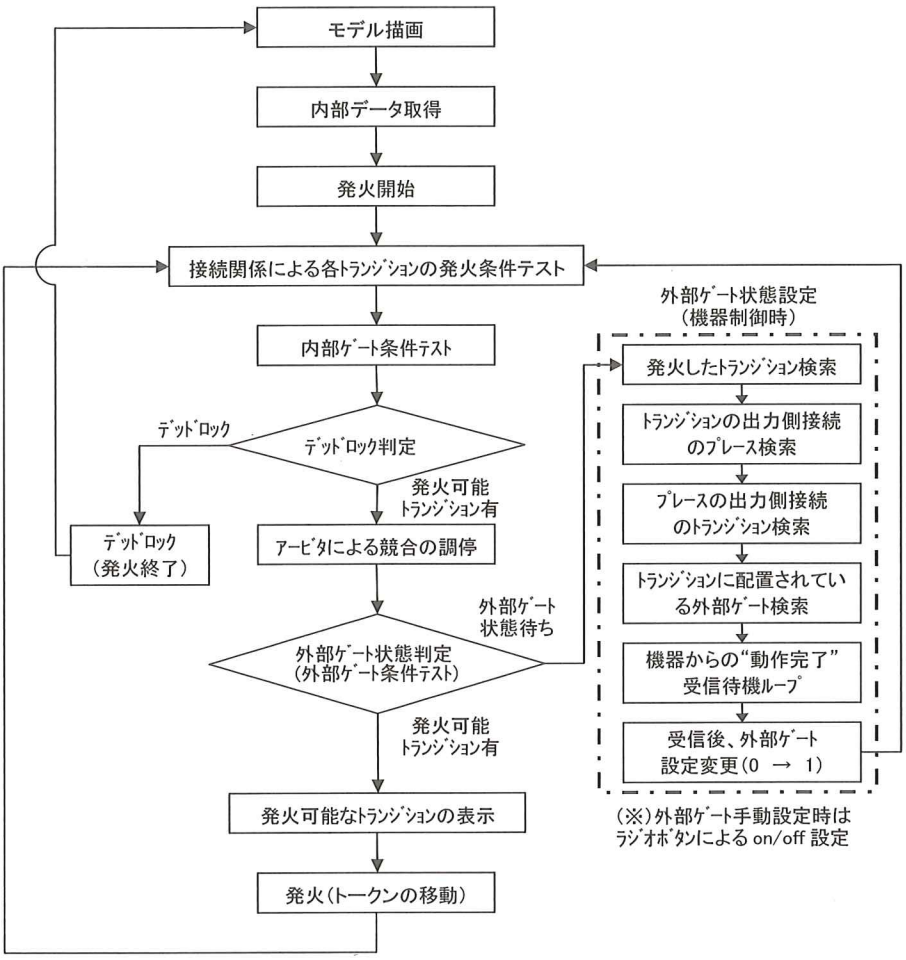


図6　シミュレーションのフローチャート

(2) 内部データ取得

モデル描画したものを接続行列，接続関係表，ゲート条件表，アービタ表，マーキング表にデータ化し保存する。

(3) 発火開始

シミュレーションまたはシミュレータによる機器の制御を開始する際に，ユーザーがボタン操作により開始を実行する。また，このとき外部ゲートの設定を手動で行う場合は，開始実行前に予め手動設定しておく必要がある。

(4) 接続関係によるトランジションの発火条件テスト

接続関係表からトランジションとプレースの接続関係を読み取り発火可能なトランジションを判定する。まず，１つのトランジションに対して接続関係にあるプレースを調べ，プレースが入力側接続のときはマーキング表によりそのプレースにトークンがなければ“０”，プレースが出力側接続のときはマーキング表によりそのプレースにトークンがあれば“０”，それ以外の場合は“１”の値をもつデータを作成する。入力側と出力側それぞれについて各プレースのデータの値の和をとり，その値がプレースの総数より少なければデータ内に“０”が存在し，それはトランジション発火不可であることを示す。全てのトランジションに対して上記を調べることで，この時点での発火可能なトランジションを表すトランジション情報表を作成する。

(5) 内部ゲート条件テスト

トランジション情報表において“１”の値をもつ発火可能なトランジションについて，内部ゲート条件が成立しているか否かを調べる。ゲート条件表より内部ゲート部分を取り出して，その値が正であれば許可ゲートであり，マーキング表により関連するプレースにトークンがなければその内部ゲートを条件にもつトランジションは発火不可である。内部ゲートの値が負であれば抑止ゲートであり，マーキング表により関連するプレースにトークンがあればその内部ゲートを条件にもつトランジションは発火不可である。これらの発火不可と判断されたトランジションに対してトランジション情報表の値を“０”に書き換える。

(6) デッドロック判定

トークンの状態により，これ以上発火ができない状況をデッドロックという。(4)，(5)の時点で発火可能なトランジションがない，すなわちトランジション情報表の値がすべてのトランジションにおいて“０”ならば，デッドロックと判定し，発火の終了を表示してシミュレーションおよび機器制御を停止する。

(7) アービタによる競合の調停

競合プレースに接続された複数のトランジションが発火可能な場合，これらのうち１つのトランジションしか発火できない。そこでアービタが一定の順序と割合で発火の権利を割り当てて調停する。アービタ表を用いて競合プレースに接続されるトランジションの番号を調べ，そのトランジションが発火可能，すなわちトランジション情報表において“１”の値をもつトランジションのとき，さらに同じプレースに接続される他のトランジションの状態を調べ，それが発火可能であるとき，そのトランジションに対してトランジション情報表の値を“０”に書き換える。今回はわかりやすく複数のトランジションの内，トランジション番号が小さい方を優先的に発火させるように作成した。このアービタのプログラムを発展させることでさらに幅広い制御が可能となる。

(8) 外部ゲート状態判定(外部ゲート条件テスト)

トランジション情報表において“１”の値をもつ発火可能トランジションについて外部ゲート条件が成立しているか否かを調べる。ゲート条件表より外部ゲート部分を取り出して，その値が正であれば許可ゲートであり，外部信号がOFFのときその外部ゲートを条件にもつトランジションは

発火不可である。外部ゲートの値が負であれば抑止ゲートであり，外部信号がONのときその外部ゲートを条件にもつトランジションは発火不可である。これらの発火不可と判断されたトランジションに対してトランジション情報表の値を“0”に書き換える。

この時点で発火可能なトランジションがなく，さらに外部ゲートが存在する場合に外部ゲート状態となる。これは外部ゲート条件テストが終わった時点でのトランジション情報表による判定であり，もし外部ゲートが存在する場合は外部信号によって発火不可と判断されるものがあると考えられるからである。また，外部ゲートが手動設定のときに外部ゲート待ちと判定された場合は外部ゲート状態設定で設定変更を促すようにしている。設定変更の際に同時にトランジション情報表を書き換えることで次回の判定時に発火可能なトランジションとする。シミュレータによる機器の制御を行う場合は，機器の動作完了を次回のトランジション発火許可条件とするために外部ゲートを用い，動作完了信号を受信したら外部ゲート設定を自動で発火許可状態に変更するようにした。

(9) 発火可能なトランジションの表示

この時点のトランジション情報表で“1”の値をもつトランジションは発火可能である。そのトランジションに対して描画時に記憶した座標データを読み出し，トランジションの色を赤色に変えることで発火可能であることを表示する。

(10) 発火（トークンの移動）

ペトリネットの発火則より発火可能なトランジションの入力側に接続されるプレース内のトークンが消滅し，出力側に接続されるプレースにトークンが生起する。(10)の時点で発火可能，すなわちトランジション情報表において“1”の値をもつトランジションに対して，その入力側に接続されるプレース内のトークンをプレースの描画時に記憶した座標データを利用してグラフィック背景色に変更する。また出力側に接続されるプレースに対しても描画時に記憶した座標データを利用してトークンを描画する。これと同時にマーキング表を書き換えて，再度(4)による発火条件テストを行うことで連続的にペトリネットのシミュレーションを実行できる。

5．シミュレータ画面の機能

ペトリネットシミュレータはパソコン（HP Compaq nx6120, CPU Pentium M 2.0GHz, OS Windows XP）上で，プログラム開発ソフトはVisual C#2005を使用して作成した。図7にペトリネットシミュレータの画面を示し，画面内の(1)～(7)の機能について説明する。

(1) モデル描画・消去機能

シミュレーションを行うためのモデル描画は必要不可欠である。ペトリネットの要素であるトランジション，プレース，アーク，ゲート，トークンに対してコマンドを設け，描画したい要素のコマンドを選択してグラフィック表示領域をマウスでクリックまたはドラッグ＆ドロップすることにより描画を行う。このとき各要素に対して接続状態や配置状態に不適正があればエラーメッセージを表示し，確実なモデル描画をできるようにしている。また，どの要素を選択・描画しようとしているかをコマンドの色替えをすることで明確にする。さらにモデルの描画時の誤りに対しては，消去機能を使用することで誤って描画した要素のみを消去することができる。消去機能には“個別消去”と“全消去”があり個別消去の場合は，消去コマンドを選択後，消去したい要素のコマンドを選択する。このときコマンドの色替えで消去する要素を明確にし，グラフィック内の要素をマウスでクリックすることにより消去を行う。全消去の場合は，全消去コマンドを選択すると確認メッセージを表示し，確認後にモデル全体の消去を行う。

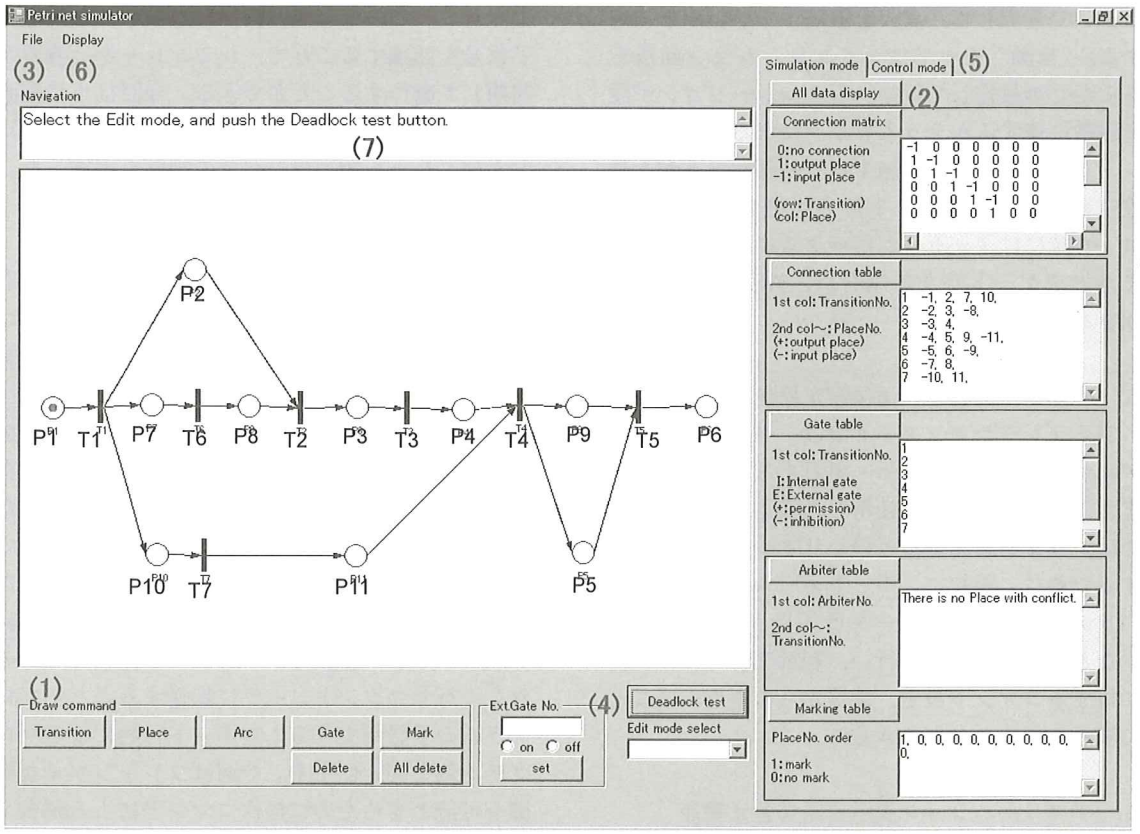


図7　ペトリネットシミュレータ画面

(2)　モデルの内部データ表示機能

接続行列，接続関係表，ゲート条件表，アービタ表，マーキング表の内部データに関して，モデル描画完了後に表示用ボタンを押して表示を行う。内部データを見ることでモデルの再確認ができ，モデルの変更や外部ゲート状態設定に役立てることができる。

(3)　モデルのデータ保存・展開機能

描画したモデルの画像データをファイルとして保存し，いつでもデータを読み出せるようにすることで効率的にシミュレーションを行うことができる。データの保存はモデル描画時に記憶した各要素のグラフィック表示領域内の座標データおよび内部データを文字列に変換してテキストファイル形式で行う。このテキストファイルはその中身を見ればある程度モデルの内容が把握できるように配慮している。データの展開はテキストファイルからペトリネット要素ごとの配置を読み取り再描画して，内部データを文字列から再びデータ化することでシミュレーションが可能な状態にする。

(4)　シミュレーション実行機能

シミュレーション時にトランジションを発火させる方法を“編集モード”で選択することができる。編集モードには“メッセージボックス”，“ラジオボタン”，“自動”があり，“メッセージボックス”モードは発火可能トランジションが存在することを示し，発火を継続するか否かをメッセージボックスで決定する。また，発火可能トランジション番号をメッセージボックスで表示する。“ラジオボタン”モードは“発火”，“終了”をラジオ

ボタンで選択して，発火を継続するか否かを決定する。“自動”モードはシミュレーション回数をあらかじめ設定して行い，シミュレーションが設定回数に達するかデッドロックになるまで自動でシミュレーションを実施する。外部ゲートが手動設定のとき，外部ゲート待ち状態およびデッドロック時にはメッセージボックスにて表示する。また外部ゲートがある場合は，外部信号のON/OFFを模擬設定する。

(5) 内部データ表示・通信設定機能

シミュレーションモードでは，ペトリネットモデルの内部データを表示・更新することができる。また，シミュレータを使用した機器の制御モードでは，さまざまな機器とのシリアル通信を可能とするために，最初に，ポートネーム，ボーレート，パリティチェック，データビット，ストップビットなどポートの設定を行い，制御実行時には機器に対するコマンド設定，機器からの返信内容などの表示を行うことができる。

(6) アプリケーション表示言語切替え機能

今後，さらにアプリケーションの内容を発展させ，シミュレータソフトとして数多くの人が利用できることを考慮し，アプリケーション内の表示言語を切替えコマンドで“日本語”および“英語”に切り替えることができる。

(7) ナビゲーション機能

モデル描画からシミュレーション終了までの必要な操作をナビゲート表示することで，スムーズにアプリケーションを使用できる。

6．ペトリネットシミュレータを用いた機器の制御

シミュレータでは機器の制御に必要な操作コマンドをプレース毎に設定でき，トランジションが発火するタイミングでシリアル通信機能を利用して機器に対して操作コマンドを送信することで制御を行う。このとき機器側から返信される動作完了信号を認識することで，シミュレータと機器が同期して動作することができる。今回は小型垂直多関節ロボット（Panasonic製PanaRobo KS-V20）およびパレット搬送コンベアの制御を実施した。

(1) ロボットの制御

PanaRoboのコントローラをコンソールモードに設定し，シリアル通信でコントローラに操作コマンドを送信することでロボットアームの動作制御が可能である[5),6)]。コマンドは文字列にCR/LFを加えた形式で送信され，コントローラ側はCR/LFを読み取ることで操作コマンドと判断し，指令を実行する。動作に関するコマンドを送信した場合は，動作完了時に“RDY”を返す。続けてコマンドを送信するときは，“RDY”を受信したことを確認後に行う必要がある。また，PanaRobo専用のロボット言語でプログラムを作成して，それを1行ずつコントローラに転送することでプログラムの登録ができる。ロボット言語によるプログラム例を図8に示す。今回はフレキシブルな制御を可能とするために操作コマンドによる制御を実施した。ロボット制御の流れを図9に示す。

PanaRoboを動かすには位置情報が必要であり，位置情報に関してもコマンドによる読出し・登録ができる。位置情報の読出しコマンドを送信した場合，位置情報データが返信される。位置情報データは，ツール中心位置の直交座標値XYZ（mm），PanaRoboのJ4軸リスト角度P（度），J5軸ロール角度R（度），座標系設定（1～4）で表される。位置情報登録をする場合は，位置番号と上記の位置情報データを含めた文字列をコマンドとして送信する。また，コントローラに付属する遠隔教示装置であるティーチングボックスを使用しても位置情報登録・編集ができる。この場合，位置情報の数値入力での登録のほかに，モーションキーでPanaRoboを動かして位置確認をしながらの登録ができる。ティーチングボックスではさらにコントローラに登録されているプログラムの実行，動作スピード設定，パラメータ（ツール長・

【プログラム例】

```
1     '#PROG = 1
2     SPEED     200,100,100
3     MOVE      F   PW 3
4     MOVE      F   L   PW 1
5     OUT        5 , 1
6     DELAY      1
7     MOVE      F   L   PW 3
8     MOVE      F   PW 2
9     MOVE      F   L   PW 4
10    OUT        5 , 0
11    DELAY      1
12    MOVE      F   L   PW 2
13    MOVE      PW 5
14    END
```

※上記プログラムはピックアンドプレース動作の例である。ここで外部出力信号は 1 をセットしたとき PanaRobo 先端ツールのハンドを閉，0 をセットしたときハンドを開とするものとする。

【プログラムの説明】

1　プログラム No. = 1
2　スピード設定　CP , PTP , RATE
　CP：スピード(mm/s),
　PTP：スピード率(%),
　RATE：設定済スピードに対する比率(%)
　※（実際のスピードは CP に PTP, RATE を乗じた値になる。CP, PTP, RATE はそれぞれ省略可能だが，その場合スピードは変化しない。最大値はそれぞれ 500, 100, 100 である。）
3　位置番号“3”に移動
　F 指定すると目標点の近傍での停止確認後，次の命令実行（以下同様）
4　位置番号“1”に直線軌跡で移動
5　外部出力信号　開始ビット“5”に 1 をセット
6　待ち時間　1×100(ms)
7　位置番号“3”に直線軌跡で移動
8　位置番号“2”に移動
9　位置番号“4”に直線軌跡で移動
10　外部出力信号　開始ビット“5”に 0 をセット
11　待ち時間　1×100(ms)
12　位置番号“2”に直線軌跡で移動
13　位置番号“5”に移動
14　プログラム終了

図 8　ロボットプログラム例（ピックアンドプレース動作）

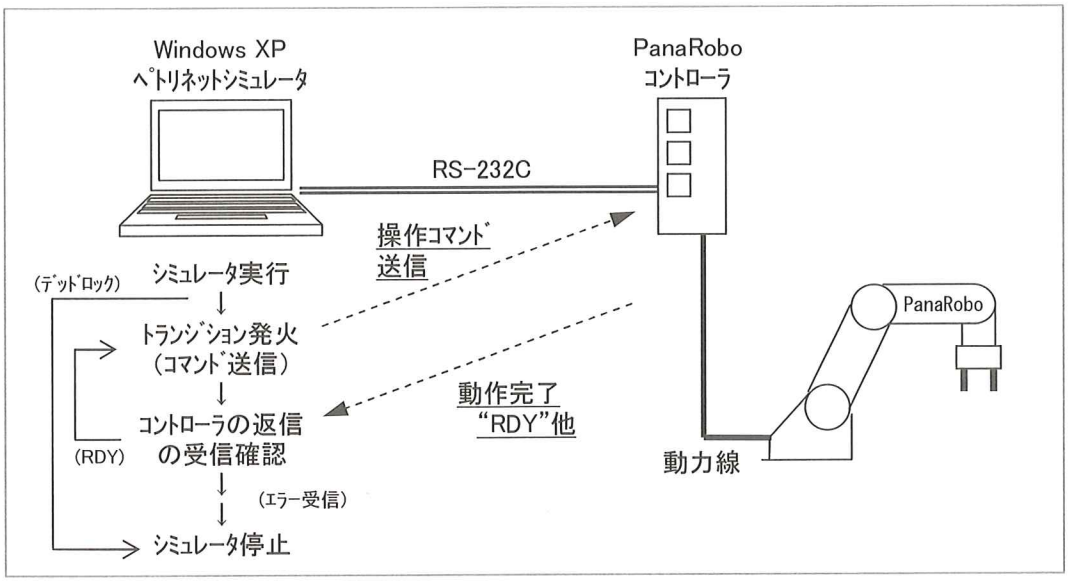


図 9　コマンド送信による PanaRobo の制御

【位置情報登録】 入力形式

PTD　1 , X = 100.00 , Y = 200.50 , Z = −50.50 , P = −90.50 , R = 80.00 , F = 1

位置情報登録コマンド（位置番号“1”）

位置情報データ（各座標・角度は小数点 2 桁までを入力）

《返信》 RDY

【位置情報読出し】 入力形式

PTR　1

位置情報読出しコマンド（位置番号“1”）

《返信》 [位置情報番号] , X = [データ] , Y = [データ] , Z = [データ] , P= [データ] ,
R = [データ] , F= [データ]

（例） 1 , X = 100.00 , Y = 200.50 , Z = −50.50 , P = −90.50 , R = 80.00 , F = 1

【指定位置移動】 入力形式

MOV　1 , 100

MOV F “位置番号” , “スピードデータ(省略可)”
※指定位置へ移動し近傍に停止後に“RDY”を返す。

移動コマンド

位置番号“1” , スピードデータ“100”(%)　（※スピードデータは省略可）

《返信》 RDY

【外部出力信号書き込み】 入力形式

DOW　0 □ □ □ □ H

外部出力信号書込みコマンド

信号データコード （外部出力の 16 ビットを 16 進の 4 桁で表す）
O:数値データを表す , □:データ(16 進コード 4 桁) , H:16 進を表す

《返信》 0 □ □ □ □ H

（例） DOW　0123AH

出力信号ビット番号

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	0
1				2				3				A			

その他 ： OPN, CLS, CAL に関してはコマンドのみ入力で返信は“RDY”となる。
DIR はコマンドのみ入力で返信は外部出力信号書込みと同様の形式で表される。

図10　PanaRobo 操作コマンド使用例

通信速度・各軸動作範囲など）編集，外部入力信号のモニタ,外部出力信号の強制出力,コントローラ内エラー読出しなどができる。今回のPanaRoboの役割はワークを移動させることであり，そのためにワークを取得する位置情報および移動先で設置する位置情報の登録が必要となる。ここで扱っている位置情報とはPanaRobo先端ツール取り付け面の中心位置の座標であり，ツールを取り付けた場合はティーチングボックスのパラメータ編集によりツール長を指定することで補正が可能である。また，PanaRoboがワークを取得および設置する際にはPanaRobo先端ツールのハンドを開閉して行うが，これはPanaRoboコントローラの外部入出力信号を利用する。主に使用するコマンドを以下に記し使用例を図10に示す。

・OPN：コンソールモードへの切替え
・CLS：パネルモードへの切替え
・CAL：PanaRobo原点位置移動（PanaRobo電源投入後，まず原点位置移動が必要）
・PTR：登録済みの位置座標読出し
・PTD：位置座標登録
・DIR：外部入力信号読出し
・DOW：外部出力信号書込み
・MOV：指定位置へ移動

“OPN”はコントローラのモードをコンソールモードに切り替えて，操作コマンドによる制御を可能な状態にする。“CLS”はコントローラのモードをパネルモードに切り替えて，コンソールモードによる制御を終了する。“CAL”はPanaRoboを操作する際，電源投入後に最初に必要な操作である原点位置移動をする。“PTR”は“MOV”にて移動する位置を指定する際に指定可能な位置番号を取得する。“PTD”はPanaRoboを移動させる位置を新規登録する。“DIR”は外部入力信号を読み出す。今回はコンベアとの連動の際に必要となるパレット位置決定完了信号の読み出しに使用する。“DOW”は，外部出力信号の強制書込みをする。今回はPanaRobo先端ツールであるハンドの開閉信号を送信するのに使用する。“MOV”は位置番号を指定してPanaRoboを目的地へ移動させるときに使用する。また，PanaRoboの移動スピードの設定も可能である。

(2)　パレット搬送コンベアの制御

シーケンサ（OMRON製　SYSMAC　CQM１）にコマンドを送信することでコンベアの制御が可能である。今回，コンベア制御に使用する主なコマンドは以下の通りである[7]。

・RR：入出力リレー状態の読出し
・KS：入出力リレー強制セット

コンベアの各センサと操作端はSYSMAC CQM１の入出力リレーCHに接続されており，“KS”を使用して操作コマンドを送信してコンベアを動作させる。レスポンスが正常であれば“RR”を使用してCH指定でコマンドを送信することでコンベアの状態を読み出して状態確認を行い，動作完了の状態になったら，次の操作コマンドを送信する。ペトリネットシミュレータを使用したコンベア制御の流れを図11に示す。

(3)　ロボットとパレット搬送コンベアの協調制御

PanaRoboとパレット搬送コンベアの協調制御のためのシステム構成例を構築し，ロボットを用いた生産システムの基本的な作業であるピックアンドプレース作業により制御システムの検証を行う。システム構成としてパソコン１台でシリアルポートを２個使用し，それぞれでPanaRoboとパレット搬送コンベアを制御する方法を考える。図12に協調制御のためのシステム構成を示す。

パレット搬送コンベアは，PanaRoboの周囲を一周するように配置されており，No.１～４コンベアおよびNo.１～３ターンテーブルによってパレットを一方向に搬送する仕様になっている。今回の検証ではNo.２コンベアの位置に作業台を配置し，作業台に加工機を模擬するシステムを考える。図13に実験システムの配置図およびロボット

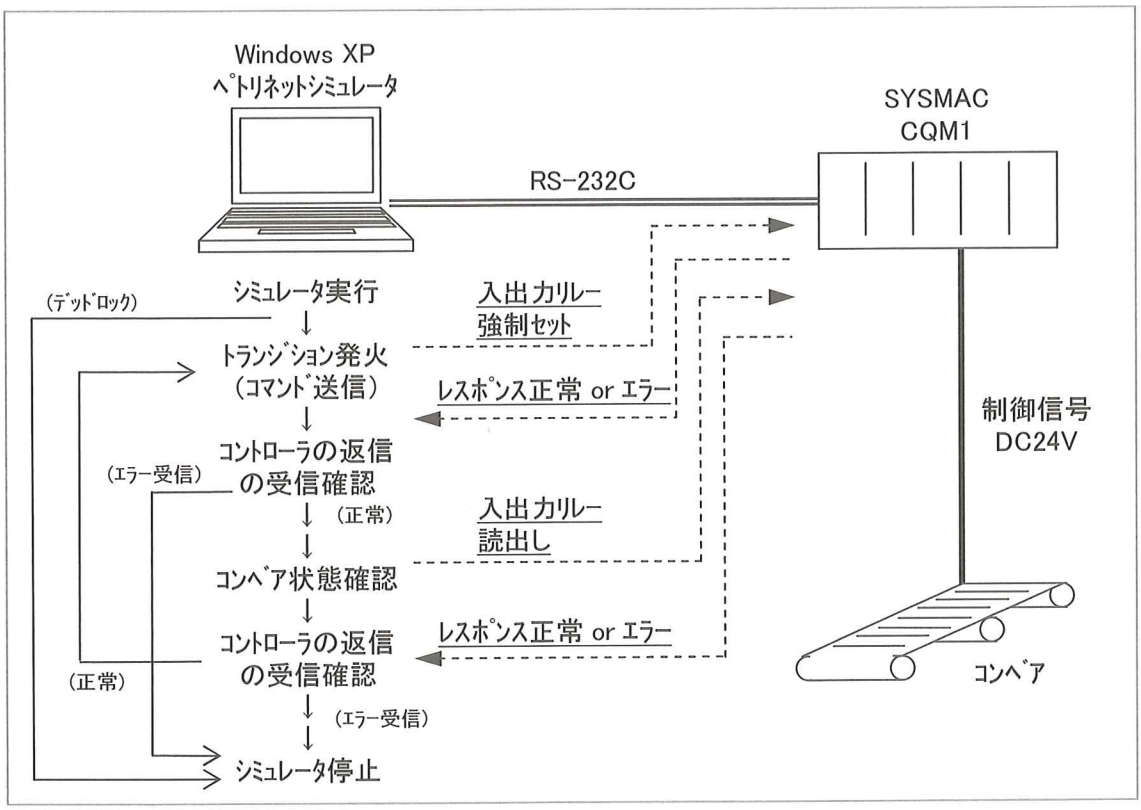


図11　コマンド送信によるパレット搬送コンベアの制御

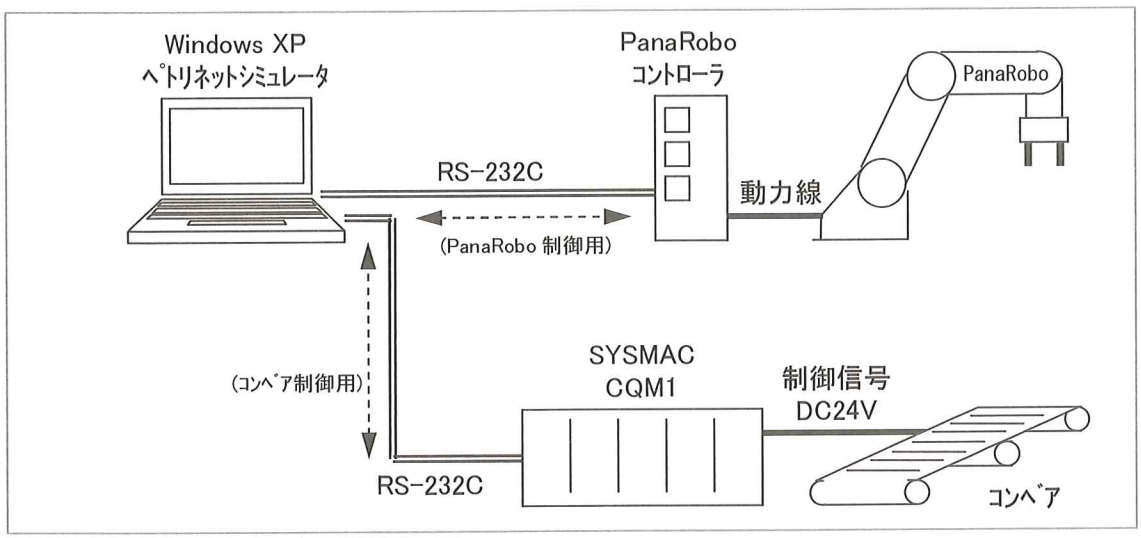
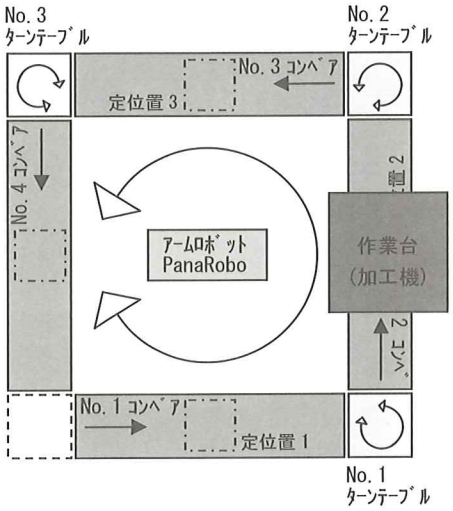


図12　PanaRobo とパレット搬送コンベアの協調制御システムの構成



① P1:No.1 ｺﾝﾍﾞｱによりﾜｰｸ搬入
② T1:定位置 1 にﾜｰｸ到着
③ P2:No.1 ｺﾝﾍﾞｱ待機
④ T2:PanaRobo のﾜｰｸ取得完了
⑤ P3:PanaRobo が作業台に移動
⑥ T3:PanaRobo が作業台に到着
⑦ P4:PanaRobo がﾜｰｸ解放
⑧ T4:PanaRobo のﾜｰｸ解放完了
⑨ P5:加工機作業中
⑩ T5:加工機の作業完了
⑪ P6:PanaRobo が作業後のﾜｰｸを取得

⑫ P7:PanaRobo が定位置1に移動
⑬ T6:PanaRobo が定位置1に到着
⑭ P8:PanaRobo がﾜｰｸを取得
⑮ P9:PanaRobo 待機

⑯ P10:加工機準備
⑰ T7:加工機準備完了
⑱ P11:加工機待機

図13　実験システム配置図および動作内容

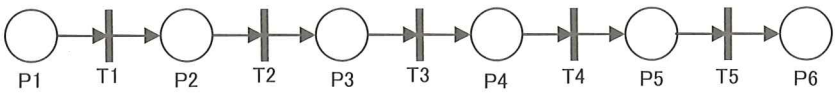


図14　ペトリネットモデル（ワークの流れ）

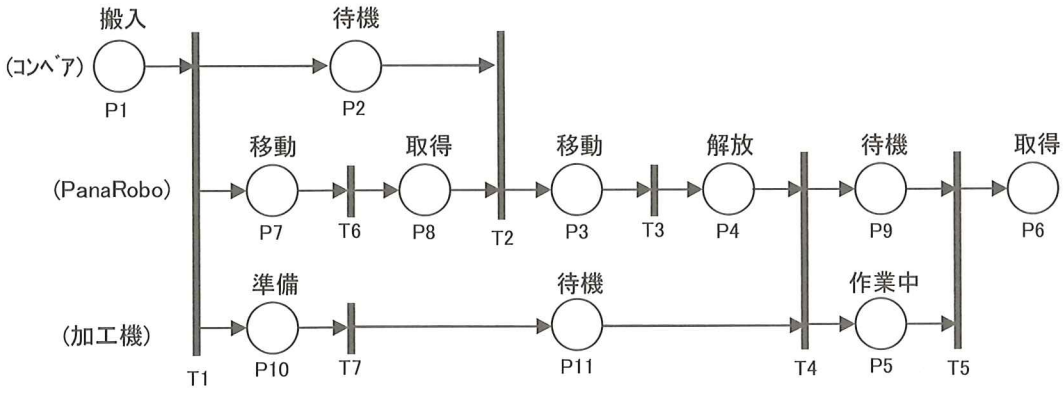
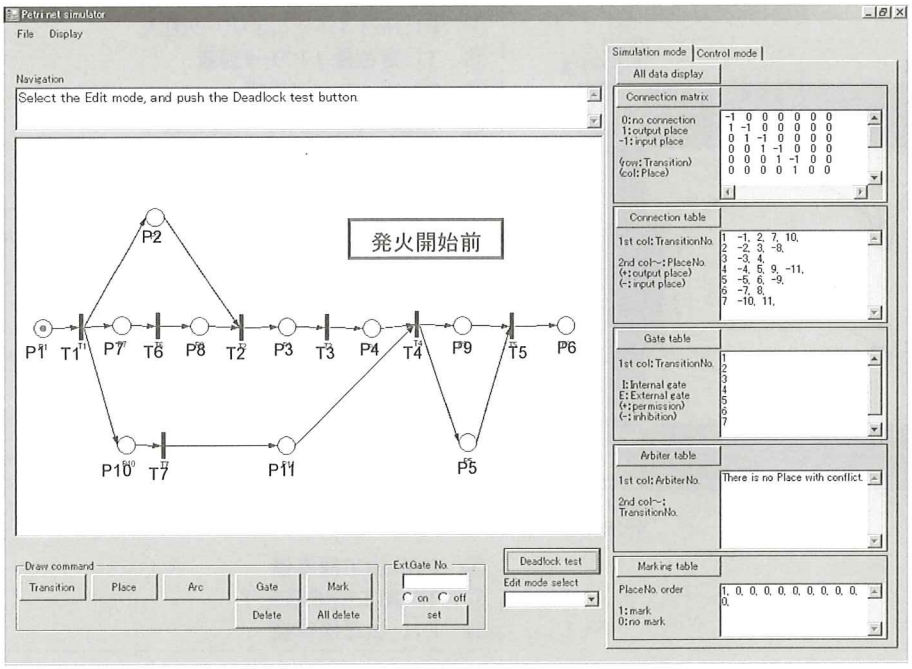


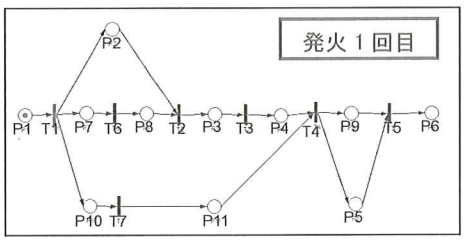
図15　ペトリネットモデル（システム全体）

による作業をペトリネットで表現する際の事象（Ti）と状況（Pi）の動作内容を示す。

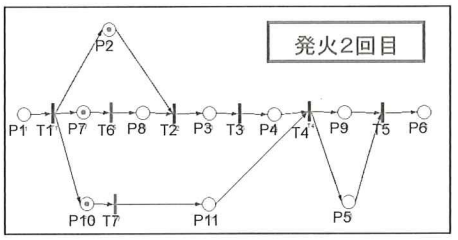
これをワークの流れに対応させてペトリネットモデルで表現すると図14のようになる。これらの動作を行うとき PanaRobo，加工機，No. 1 コンベアが使用されるのでそれらの動作を付加したモ


Petri net simulator
File Display
Navigation
Select the Edit mode, and push the Deadlock test button.
発火開始前
Simulation mode
Control mode
All data display
Connection matrix
Connection table
Gate table
Arbiter table
Marking table
Draw command
Transition
Place
Arc
Gate
Mark
Delete
All delete
Ext.Gate No.
set
Deadlock test
Edit mode select

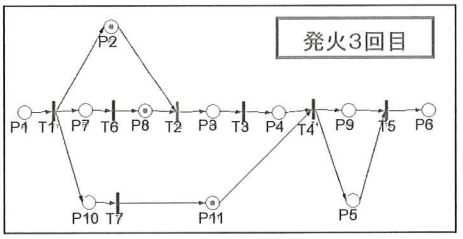
(1)


発火１回目

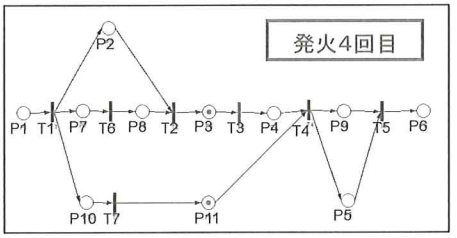
(2)


発火2回目

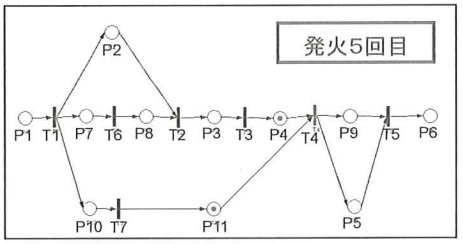
(3)


発火3回目

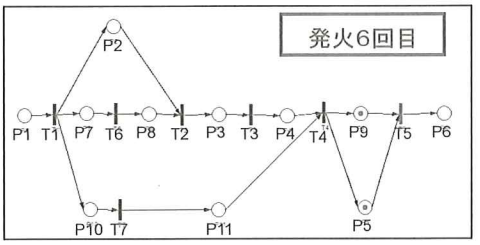
(4)


発火4回目

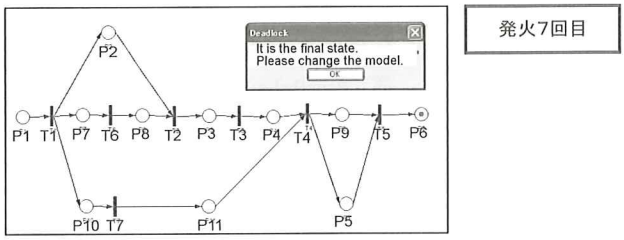
(5)


発火5回目

(6)


発火6回目

(7)



(8)

図16　シミュレーション結果

デルを図15に示す。

図15のペトリネットモデルにおいて，まずNo. 1コンベアがワークを乗せたパレットを搬入し定位置1に到着すると，PanaRoboが定位置1に向かい移動を始め，加工機は作業台にて作業準備を始める。PanaRoboは定位置1に移動完了後，パレット上のワークを取得開始する。No. 1コンベアが定位置1で待機中にPanaRoboによりワークが取得されると取得完了し，PanaRoboはワークを掴んだまま作業台へ移動する。PanaRoboが作業台に到着すると作業台上にワークを解放開始し，加工機が準備完了して待機中であれば解放完了する。その後，加工機により作業開始しPanaRoboは待機状態になる。加工機による作業が完了すると，ワークはPanaRoboにより取得され作業台から搬出される。このペトリネットモデルをシミュレーションし，所要の挙動を確認できた。シミュレータによる挙動を図16に示す。

7．制御実験結果の検証

ペトリネットシミュレータによるシステムの挙動シミュレーションの結果，図16に示すようにトークンがペトリネットの発火則に基づき移動することを確認した。その後，シミュレータを用いたPanaRobo単体の制御を実施した。プレース毎に操作指令コマンドを割り当て，シミュレーションをメッセージボックスまたはラジオボタンによる手動モードで実行し，外部ゲートも手動で設定するとトランジションの発火とともにPanaRoboは指定の作業位置へ移動を開始し，モデル通りの動作を確認できた。

次にPanaRoboの制御を動作状況を監視しながら自動で行うことを考え，作業開始後に得られるPanaRoboからの動作完了信号を外部ゲート信号にプログラムで変換し，発火ループ中に実行される“外部ゲート条件テスト”によりトランジション発火の条件に含めた。操作指令コマンド送信後に，図6に示したフローチャート内の“外部ゲート状態設定”を実施することでシミュレータは動作完了信号を受信するまで待機状態となり，受信後にトランジション発火してシミュレータとPanaRoboの同期動作を確認できた。なお，図6のシミュレータのフローチャートにおいて，“外部ゲート条件テスト”は“外部ゲート状態判定”内に含めている。“アービタによる競合の調停”の調停方法次第では外部ゲート条件テストをその前に実施する必要もでてくる。例えば，複数の機器がありその中で優先順位が決まっている場合は先にアービタによる競合の調停を実施できるが，今回採用しているトランジションNo.が少ない方が優先としている場合は外部ゲート条件テストを先に実施しないと結果が変わってくる。以上によりシミュレーションでの挙動とロボット単体での制御を確認できたので，ロボットとパレット搬送コンベアのシステムで実施したときも協調制御が達成できることは容易に予想できる。

8．結　　言

ペトリネットシミュレータを新たに開発し，そ

のシミュレータを用いた生産ラインの統合制御を目的としてPCを用いた汎用の制御システムを製作し，実験結果の検証を行った。生産システムにおいてロボットのような知能化機器のフレキシブルな制御を行う場合は，通信による機器の状態把握が確実な制御のためにも不可欠であり，外部機器の動作条件を付加したペトリネットモデルのシミュレータをコントローラの一部として組み込むことは重要な意義がある。今回は，生産システムの要素として産業用ロボット一台と搬入，搬出用パレット搬送コンベア各一台のみを使用したが，ロボットの台数を増やし，部品の到着頻度等，実際の生産システムにおける条件のもとでアービタの調停ルールを含む制御ソフトウェアの検証を行う予定である。また，ロボットや外部機器の台数が増えると制御用コンピュータも一台でなく，複数台による分散処理が有効と考えられる。そのときのコンピュータ間の接続や信号のやり取り，同期制御の方法の検討が必要である。さらに大規模かつ複雑な生産システムの設計・解析や制御においては，その大きさゆえに階層化が必要となってくる。まずマクロに全体システムをとらえ，マクロにとらえたある一つの部分をまた一つのシステムとみなし分解していくことは，ペトリネットのある一つのプレースの中をのぞくとまたそれがペトリネットになっていることに対応づけられる。このようなものを扱えるようにシミュレータとコントローラを拡張することを計画中である。

参考文献

1）W.ライシッヒ（長谷川健介，高橋宏治訳）：ペトリネット理論入門，シュプリンガーフェラーク東京，1988
2）村田忠夫：ペトリネットの解析と応用，近代科学社，1992
3）長谷川健介，高橋宏治，増田良介，大野秀嶺：非連続システム制御のためのマーク流れ線図の提案，計測自動制御学会論文集，Vol. 20，No. 2，122-129，1984
4）徳重秀二，山本信行：ペトリネットシミュレータの開発，長崎総合科学大学機械工学科情報制御工学コース卒業論文，1987
5）松下電器産業㈱ロボット・FAシステム事業部：PanaRobo KS-V20　取扱説明書
6）松下電器産業㈱ロボット・FAシステム事業部：PanaRobo KS-PC　取扱説明書
7）OMRON：プログラマブルコントローラ SYSMAC　C60H　ユーザーズマニュアル RS-232C インタフェース編